Memory-Mapper-Verwaltung mittels XBIOS

Folgende Funktionen werden bei MSX-DOS 2 über BIOS-Aufrufe abgewickelt:

- Feststellen der vorhandenen Mapper-Kapazität
- Bereitstellen von 16K-Blöcken
- Schreiben und Lesen von Einzelbytes (analog zu RDSLT/WRSLT)
- Einschalten von Mappersegmenten (analog zu ENASLT)
- Feststellen der momentan eingeschalteten Seite
- Unterprogrammaufrufe in Mappersegmente (analog zu CALSLT)

Wie geht's?

Als erstes muß festgestellt werden, ob überhaupt ein XBIOS existiert. Diese Information finden wir in der Speicherstelle mit der Adresse $FB20_h$. Ist der Wert dieses Bytes ungerade (Bit 0 gesetzt), haben wir einen XBIOS-Einsprung auf Adresse $FFCA_h$, ist er gerade (Bit 0 gelöscht), haben wir keinen.

Damit sind wir aber noch nicht fertig. Das XBIOS kann auch z.B. zu einer RS-232-Karte gehören. Deshalb wird beim Aufruf eine Geräte-Kennung im D-Register übergeben; die Kennung für den Mapper ist 04_h.

Im E-Register übergeben wir die Nummer der auszuführenden Funktion. Hier gibt es zwei Möglichkeiten:

01_h:	Mapper-Belegung feststellen
02_h:	Adresse der BIOS-Sprungleiste feststellen

Mapper-Belegung

XBIOS-Funktion 01_h liefert folgende Werte zurück:

A-Register:	Slotnummer des "Master-Mappers" (TPA-Slot)
HL-Register:	Adresse der Belegungstabelle

Die Belegungstabelle enthält für jeden Slot, in dem sich ein Memory-Mapper befindet, einen 8 Byte langen Eintrag mit folgendem Inhalt:

+0	Slotnummer
+1	Anzahl der Mapper-Segmente in diesem Slot
+2	Anzahl der freien Segmente in diesem Slot
+3	Anzahl der System-Segmente in diesem Slot
+4	Anzahl der von Anwenderprogrammen belegten Segmente in diesem Slot
+5..+7	Unbenutzt (Null)
+8	Slotnummer des nächsten Mappers usw.

Das Ende der Tabelle wird durch eine 0 im ersten Byte (anstelle der Slotnummer) gekennzeichnet. Beispiel für eine Belegungstabelle:

Memory-Mapper-Verwaltung mittels XBIOS

02 20 10 06 0A 00 00 00 8B 08 00 08 00 00 00 00 00

Ende der Tabelle

Mapper in Slot 3-2
insgesamt 8 Segmente (128K)
Alle Segmente vom Betriebssystem belegt.

Mapper in Slot 2
Insgesamt 32 Segmente (512K)
16 Segmente frei
6 Segmente vom Betriebssystem belegt
10 Segmente von Anwenderprogrammen belegt

Sprungtabelle

XBIOS-Funktion 02_h liefert folgende Werte:

A-Register: Anzahl der Mappersegmente im TPA-Slot
B-Register: Nummer des TPA-Slots
C-Register: Anzahl der freien Mappersegmente im TPA-Slot
HL-Register: Adresse der Sprungtabelle

Adresse	Name	Funktion
(HL+00_h)	ALL_SEG	Mapper-Segment reservieren
(HL+03_h)	FRE_SEG	Mapper-Segment freigeben
(HL+06_h)	RD_SEG	Ein Byte aus einem Mapper-Segment lesen
(HL+09_h)	WR_SEG	Ein Byte in ein Mapper-Segment schreiben
(HL+$0C_h$)	CAL_SEG	Unterprogramm in Mapper-Segment aufrufen
(HL+$0F_h$)	CALLS	Unterprogramm in Mapper-Segment aufrufen (Adresse "inline")
(HL+12_h)	PUT_PH	Mapper-Segment einschalten, Adresse in HL
(HL+15_h)	GET_PH	Mapper-Segment ermitteln, Adresse in HL
(HL+18_h)	PUT_P0	Mapper-Segment im Bereich 0000_h-$3FFF_h$ einschalten
(HL+$1B_h$)	GET_P0	Eingeschaltetes Mapper-Segment im Bereich 0000_h-$3FFF_h$ ermitteln
(HL+$1E_h$)	PUT_P1	Mapper-Segment im Bereich 4000_h-$7FFF_h$ einschalten
(HL+21_h)	GET_P1	Eingeschaltetes Mapper-Segment im Bereich 4000_h-$7FFF_h$ ermitteln
(HL+24_h)	PUT_P2	Mapper-Segment im Bereich 8000_h-$BFFF_h$ einschalten
(HL+27_h)	GET_P2	Eingeschaltetes Mapper-Segment im Bereich 8000_h-$BFFF_h$ ermitteln
(HL+$2A_h$)	PUT_P3	Keine Funktion (der Speicherbereich von $C000_h$-$FFFF_h$ darf nicht umgeschaltet werden, weil sich hier die Systemvariablen befinden)
(HL+$2D_h$)	GET_P3	Eingeschaltetes Mapper-Segment im Bereich $C000_h$-$FFFF_h$ ermitteln

Die BIOS-Routinen aus der Sprungleiste

(HL+00h) ALL_SEG Mapper-Segment reservieren

rein:	A	00: automatische Freigabe (FRE_SEG) bei Programm-Ende 01: für Systemerweiterung, keine automatische Freigabe
	B	Slotnummer (00 = TPA-Slot) + Funktionscode Funktionscodes: 00: Segment im angegebenen Slot reservieren 10: Segment in beliebigem Slot reservieren, aber *auf keinen Fall* im angegebenen Slot 20: Segment *nach Möglichkeit* im angegebenen Slot reservieren; wenn da kein Platz mehr ist, in einem beliebigen anderen Slot 30: Segment in beliebigem Slot reservieren, aber *nur dann* im angegebenen Slot, wenn alle anderen ausgebucht sind
	SP	*nicht* im Bereich von 8000h bis BFFFh!
raus:	Carry	gesetzt: Kein freies Segment gefunden gelöscht: Ok
	A	Nummer des reservierten Segments
	B	Nummer des Slots, in dem sich der Mapper befindet (00 = TPA)

(HL+03h) FRE_SEG Mapper-Segment freigeben

rein:	A	Nummer des Segments
	B	Slotnummer des Segments (00 = TPA-Slot)
	SP	*nicht* im Bereich von 8000h bis BFFFh!
raus:	Carry	gesetzt: Fehler gelöscht: Ok

ACHTUNG: Nur Segmente freigeben, die wirklich mit ALL_SEG reserviert worden sind! Das Betriebssystem wehrt sich nicht dagegen, Teile der RAM-Disk, die TPA oder sich selbst der Allgemeinheit zur freien Verfügung zurückzugeben - mit allen zu erwartenden Folgen...

(HL+06h) RD_SEG Ein Byte aus einem Mapper-Segment lesen

rein:	A	Nummer des Segments
	HL	Adresse des Bytes
	SP	*nicht* im Bereich von 8000h bis BFFFh!
raus:	A	gelesenes Byte

Memory-Mapper-Verwaltung mittels XBIOS

(HL+09_h) WR_SEG — Ein Byte in ein Mapper-Segment schreiben

rein:	A	Nummer des Segments
	E	zu schreibendes Byte
	HL	Adresse des Bytes
	SP	*nicht* im Bereich von 8000_h bis $BFFF_h$!

(HL+$0C_h$) CAL_SEG — Unterprogramm in Mapper-Segment aufrufen

rein:	IY	Nummer des Segments im High-Byte
	IX	Adresse des Unterprogramms
	andere	Was das Unterprogramm halt braucht
raus:	Register je nach Unterprogramm	

(HL+$0F_h$) CALLS — Unterprogramm in Mapper-Segment aufrufen

rein: Segmentnummer und Adresse "inline":

```
CALL    CALLS
DB      segment
DW      adresse
```

Register je nach Unterprogramm

raus: Register je nach Unterprogramm

(HL+12_h) PUT_PH — Mapper-Segment einschalten

rein:	A	Nummer des Segments
	HL	Adresse des umzuschaltenden Bereichs

(HL+15_h) GET_PH — Mapper-Segment ermitteln

rein:	HL	Adresse des fraglichen Speicherbereichs
raus:	A	Nummer des gerade aktiven Segments

(HL+18_h) PUT_P0 — Mapper-Segment im Bereich 0000_h-$3FFF_h$ einschalten

rein:	A	Nummer des Segments

(HL+$1B_h$) GET_P0 — Aktives Segment im Bereich 0000_h-$3FFF_h$ ermitteln

raus:	A	Nummer des Segments

(HL+$1E_h$) PUT_P1 — Mapper-Segment im Bereich 4000_h-$7FFF_h$ einschalten

rein:	A	Nummer des Segments

(HL+21_h) GET_P1 Aktives Segment im Bereich 4000_h-$7FFF_h$ ermitteln

raus: A Nummer des Segments

(HL+24_h) PUT_P2 Mapper-Segment im Bereich 8000_h-$BFFF_h$ einschalten

rein: A Nummer des Segments

(HL+27_h) GET_P2 Aktives Segment im Bereich 8000_h-$BFFF_h$ ermitteln

raus: A Nummer des Segments

(HL+$2A_h$) PUT_P3 Keine Funktion

(HL+$2D_h$) GET_P3 Aktives Segment im Bereich $C000_h$-$FFFF_h$ ermitteln

raus: A Nummer des Segments

Allgemeine Bemerkungen

Beim Hantieren mit Mapper-Segmenten sind die üblichen, von den Systemaufrufen ENASLT, CALSLT usw. bekannten Regeln einzuhalten:

- Der Stack darf nicht im umgeschalteten Bereich liegen,
- wenn im unteren Adreßbereich (0000_h-$3FFF_h$) umgeschaltet wird, müssen die Interrupts ausgeschaltet oder vorher die Base-Page (0000_h-$00FF_h$) in das neue Mapper-Segment kopiert worden sein

Die BIOS-Routinen zur Mapperverwaltung bedienen *nur den Mapper*; wenn auf einen Mapper in einem anderen Slot zugegriffen werden soll, ist das Anwenderprogramm für die entsprechenden ENASLT-Aufrufe verantwortlich. Die Reihenfolge ist dabei nicht entscheidend: Um ein Unterprogramm in einem externen Mappersegment aufzurufen, kann entweder zunächst der Slot mit ENASLT umgeschaltet und dann das Unterprogramm mit CAL_SEG angesprungen oder erst der Mapper mit PUT_Px umgeschaltet und dann das Unterprogramm mit CALSLT angesprungen werden. Nicht vergessen, anschließend wieder den normalen Speicher einzuschalten! (Die Nummer des TPA-Slots wird von XBIOS-Funktion 01_h in A und von XBIOS-Funktion 02_h in B zurückgegeben.)

Beispiel

Das folgende Programm reserviert ein Mapper-Segment (vereinfachtes Verfahren = auch bei Mehrfach-Mappern nur im TPA-Slot), lädt ein Unterprogramm hinein,

ruft es dort auf und gibt schließlich den Mapper wieder frei.

Memory-Mapper-Verwaltung mittels XBIOS


```
                              TITLE   MAPDEMO1

                C             $INCLUDE DOS2LB
                C             ; Makros und EQUates für DOS2
                C
                C             .REQUEST DOS2LB
                C             .LIST

 0000'                        ASEG
                              ORG     100H

 0100                 DEMO1:  ; Platz genug für Stack?

 0100   2A 0006               LD      HL,(MSXDOS+1)   ; TPA-Größe
 0103   7C                    LD      A,H
 0104   FE C1                 CP      0C1H            ; mindestens bis C100
 0106   DA 0182               JP      C,NOMEM
 0109   F9                    LD      SP,HL           ; Stack umsetzen


                              ; Nachsehen, ob Mapper verfügbar

 010A   CD 0175               CALL    ISMAP?          ; liefert Anzahl der freien Segmente in C
 010D   0D                    DEC     C               ; wenn C Null war, ist es jetzt -1
 010E   FA 0182               JP      M,NOMEM         ; Mapper voll -) Fehler


                              ; BIOS-Sprungleiste anfertigen

 0111   11 018A               LD      DE,ALL_SEG
 0114   01 0030               LD      BC,3*16
 0117   ED B0                 LDIR                    ; HL stammt noch von "ISMAP?"


                              ; ein Mapper-Segment reservieren

 0119   AF                    XOR     A               ; für Anwenderprogramm
 011A   47                    LD      B,A             ; nur im TPA-Slot
 011B   CD 018A               CALL    ALL_SEG         ; Segment reservieren
 011E   32 01BA               LD      (SEGMENT),A
 0121   DA 0182               JP      C,NOMEM         ; Fehler -) meckern


                              ; Mapper einschalten (Adreßbereich 4000-7FFF)

 0124   CD 01AB               CALL    GET_P1          ; momentanes Segment merken
 0127   32 01BB               LD      (NORMSEG),A

 012A   3A 01BA               LD      A,(SEGMENT)
 012D   CD 01A8               CALL    PUT_P1          ; Segment umschalten
```

```
MAPDEMO1        MACRO-80 3.43   03-Jul-89       PAGE    1-1


                                    ; Unterprogramm in den Mapper kopieren

  0130    21 015E                   LD      HL,UPRO
  0133    11 4000                   LD      DE,TESTPRG
  0136    01 0017                   LD      BC,PROGLEN
  0139    ED B0                     LDIR


                                    ; Mapper wieder ausschalten

  013B    3A 01BB                   LD      A,(NORMSEG)
  013E    CD 01A8                   CALL    PUT_P1


                                    ; Hier kann jetzt das Programm normal weitermachen
                                    ; und Über die gesamte TPA verfügen
                                    ;
                                    ; ...
                                    ; ...
                                    ; ...


                                    ; Unterprogramm im Mapper aufrufen

  0141    FD 2A 01B9                LD      IY,(SEGMENT-1)
  0145    DD 21 4000                LD      IX,TESTPRG
  0149    CD 0196                   CALL    CAL_SEG


                                    ; und weiter gehts im Programm
                                    ;
                                    ; ...
                                    ; ...
                                    ; ...


                                    ; Mapper-Segment wieder freigeben (Ordnung muß sein)
                                    ; Eigentlich nicht nötig, denn bei Programm-Ende wird
                                    ; der reservierte Mapperspeicher vom Betriebssystem wieder
                                    ; freigegeben. Das klappt aber nur, wenn das Programm als
                                    ; COMMAND2-Prozeß läuft und nicht, wenn es z.B. direkt
                                    ; von Wordstar mit "Run a program" gestartet wird.

  014C    3A 01BA                   LD      A,(SEGMENT)
  014F    06 00                     LD      B,0             ; TPA-Slot
  0151    CD 018D                   CALL    FRE_SEG
  0154    06 DF                     LD      B,.INTER
  0156    DA 0184                   JP      C,EXIT          ; Fehler beim Freigeben???


                                    ; Normales Programm-Ende

  0159    06 00                     LD      B,0
  015B    C3 0184                   JP      EXIT
```

```
                                ; Unser Beispiel-Unterprogramm
                                ; Es gibt nur den Text "Hier bin ich!" aus.

 015E                   UPRO:   .PHASE 4000H            ; Soll auf dieser Adresse laufen

 4000   11 4009         TESTPRG:LD      DE,MESSAGE
 4003   0E 09                   LD      C,_STROUT
 4005   CD 0005                 CALL    MSXDOS
 4008   C9                      RET

 4009   48 69 65 72     MESSAGE:DB      'Hier bin ich!$'
 400D   20 62 69 6E
 4011   20 69 63 68
 4015   21 24

                                .DEPHASE
 0017                   PROGLEN EQU     $-UPRO


                                ; Unterprogramm "Mapper verfügbar?"
                                ; es wird nur der Mapper im Hauptslot geprüft!
                                ;
                                ; raus: C       Anzahl der freien Mappersegmente
                                ;       wenn C <> 0:
                                ;       A       Gesamtzahl der Mappersegmente im Slot
                                ;       B       Nummer des TPA-Slots
                                ;       HL      Adresse der BIOS-Sprungleiste
                                ;

 0175   0E 00           ISMAP?: LD      C,0             ; Stell dir vor, es ist kein Mapper da
 0177   3A FB20                 LD      A,(HOKVLD)      ; BIOS?
 017A   0F                      RRCA
 017B   11 0402                 LD      DE,0402H        ; Mapper-Funktionsnummer
 017E   DC FFCA                 CALL    C,EXTBIO        ; kein XBIOS -> kein Mapper, sonst aufrufen
 0181   C9                      RET


                                ; Fehlermeldung "Not enough memory"

 0182   06 DE           NOMEM:  LD      B,.NORAM
 0184   0E 62           EXIT:   LD      C,_TERM
 0186   CD 0005                 CALL    MSXDOS
 0189   C7                      RST     0


                                ; BIOS-Sprungleiste (Kopie)

 018A                   ALL_SEG:DS      3
 018D                   FRE_SEG:DS      3
 0190                   RD_SEG: DS      3
 0193                   WR_SEG: DS      3
 0196                   CAL_SEG:DS      3
 0199                   CALLS:  DS      3
 019C                   PUT_PH: DS      3
 019F                   GET_PH: DS      3
```

```
 01A2                   PUT_P0: DS      3
 01A5                   GET_P0: DS      3
 01A8                   PUT_P1: DS      3
 01AB                   GET_P1: DS      3
 01AE                   PUT_P2: DS      3
 01B1                   GET_P2: DS      3
 01B4                   PUT_P3: DS      3
 01B7                   GET_P3: DS      3

 01BA                   SEGMENT:DS      1               ; unser Mapper-Segment
 01BB                   NORMSEG:DS      1               ; normales TPA-Segment

                                END     DEMO1


No Fatal error(s)
```